REMARKS/ARGUMENTS

Claims 1 - 8 are pending.

Claims 2, 4, 6, and 8 were rejected under 35 U.S.C. Section 112, Second Paragraph.

Claims 1 - 8 were rejected under 35 U.S.C. Section 102 for allegedly being anticipated by Klauser et al., Dynamic Hammock Predication for Non-predicated Instruction Set Architecture, October 1998.

As requested, the specification has been reviewed for minor errors. Counsel for Applicant is not aware of any errors that require correction.

Claims 2, 4, 6, and 8 have been amended to more clearly recite the subject matter of the present invention. The Section 112 rejection is believed to be overcome.

Counsel appreciates the Examiner's detailed reply to the earlier-filed response mailed August 19, 2004. As argued then, and re-presented herein, a distinction between Klauser et al. and the present invention as recited in the pending claims is the clear need for compilation in their dynamic predication architecture. "In this paper, we concentrate on dynamic predication for simple *branch hammocks*... . In this work, we assume the conditional branch starting a predicatable hammock... would be marked by a compiler or binary instrumentation tool." *Col. 2, beginning of 2nd full paragraph, and last full paragraph.* Thus, Klauser et al. must recompile the program code in order to produce "marked" code. By contrast, an important benefit of a computer device according to the present invention is the absence of any need to re-compile program code.

The Examiner explained there were no limitations in the claims relating to the lack of a compiler. *O.A., page 6, paragraph 22.* In response, the independent claims have been amended to more clearly recite this aspect of the present invention. Independent claim 1, for example, is directed to "machine code that is executable by said computing device, said machine code also being executable by a target computing device different from said computing device." The claim is directed to machine code, "wherein said machine code is executable by said computing device without recompiling, so that the same machine code is executable by said

target computing device and by said computing device." A computing device according to the present invention therefore can provide "hardware conversion of control flow in machine code" without having to perform a recompile operation. Consequently, the predicate assignment and predicate usage are "invisible to [the] instruction set architecture" of the target computer device.

On this last point, the Examiner explained that "the method of Klauser is, in fact, 'invisible to the instruction set architecture' and 'the user.'" *O.A., page 6, paragraph 23.* The Examiner notes that in Klauser "a compiler is used to 'mark' the conditional branches for easy identification," and concludes that this "is no way a modification to the existing instruction set architecture." With all due respect, the Examiner is in error.

An instruction set architecture refers to the structure of the machine code which specifies the bits used to define the opcodes, how the registers are specified, how immediate values are identified, and so on. By using a compiler to "mark" the code, Klauser et al. insert additional information in the machine code that is produced by their compilation process. In other words, the structure of the machine code is enhanced to accommodate "marking" the machine code. In fact, Klauser et al. show in Fig. 3 a "decode unit" that includes an instruction register (IR) that produces a signal "hammock branch". Their instruction register is able to produce such a signal because the machine code they produce includes additional information to signify a "hammock branch." In other words, the instruction set architecture is modified. Thus, in order to use the Klauser et al. computing device, the source code of an application must be recompiled to produce machine code corresponding to their device. In stark contrast, a computing device according to the present invention does not require re-compiling, and this is emphasized in the claims as amended.

The Examiner notes that "Klasuser's conversion of control flow to predicates is performed at execution time" and hence is said to operate on an instruction set without support for predication. *O.A., page 6, paragraph 23, citing column 2, lines 3 -5 in Klauser et al.* It is true that Klauser et al. perform predication at execution time. However, they are able to do this by compiling the code to produce machine code that is "marked" to indicate branches. Their compiling produces machine code that their decode unit (Fig. 3) can understand; the machine

code is different from the original instruction set architecture. So, Klauser et al. is applicable to an instruction set without support for predication because they produce machine code having a different instruction set architecture, one that provides "marking" to facilitate their predication technique. By comparison, a computing device according to the present invention does not require re-compilation.

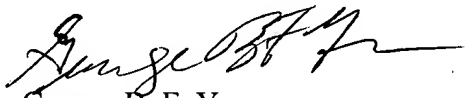
The distinction is indeed subtle, and it is proposed that a telephonic discussion with the inventor may be helpful to clarify the distinction between what is taught in Klauser et al. and what is recited in the pending claims.

CONCLUSION

In view of the foregoing, Applicant believes all claims now pending in this Application are in condition for allowance and an action to that end is respectfully requested.

If the Examiner believes a telephone conference would expedite prosecution of this application, please telephone the undersigned at 650-326-2400.

Respectfully submitted,



George B. F. Yee
Reg. No. 37,478

TOWNSEND and TOWNSEND and CREW LLP
Two Embarcadero Center, Eighth Floor
San Francisco, California 94111-3834
Tel: 650-326-2400
Fax: 415-576-0300
GBFY:deh:cmm
60350962 v1